

## Ruby master - Feature #16557

### Deduplicate Regexp literals

01/23/2020 11:49 AM - byroot (Jean Boussier)

<b>Status:</b>	Open
<b>Priority:</b>	Normal
<b>Assignee:</b>	
<b>Target version:</b>	

#### Description

Pull Request: <https://github.com/ruby/ruby/pull/2859>

#### Context

Real world application contain many duplicated Regexp literals.

From a rails/console in Redmine:

```
>> ObjectSpace.each_object(Regexp).count
=> 6828
>> ObjectSpace.each_object(Regexp).uniq.count
=> 4162
>> ObjectSpace.each_object(Regexp).to_a.map { |r| ObjectSpace.memsize_of(r) }.sum
=> 4611957 # 4.4 MB total
>> ObjectSpace.each_object(Regexp).to_a.map { |r| ObjectSpace.memsize_of(r) }.sum - ObjectSpace.ea
ch_object(Regexp).to_a.uniq.map { |r| ObjectSpace.memsize_of(r) }.sum
=> 1490601 # 1.42 MB could be saved
```

Here's the to 10 most duplicated regexps in Redmine:

```
147: /"/
107: /\s+/
103: //
89: /\n/
83: /'/
76: /\s+/m
37: /\d+/
35: /\[/
33: ./
33: /\./
```

Any empty Rails application will have a similar amount of regexps.

#### The feature

Since <https://bugs.ruby-lang.org/issues/16377> made literal regexps frozen, it is possible to deduplicate literal regexps without changing any semantic and save a decent amount of resident memory.

#### The patch

I tried implementing this feature in a way very similar to the frozen\_strings table, it's functional but I'm having trouble with a segfault on Linux: <https://github.com/ruby/ruby/pull/2859>

#### History

#1 - 01/23/2020 01:36 PM - Eregon (Benoit Daloze)

- Description updated

#2 - 01/23/2020 01:38 PM - Eregon (Benoit Daloze)

This is quite interesting, and would also avoid compiling these duplicated Regexp again, which likely saves quite a bit of startup time.

#3 - 01/23/2020 03:37 PM - byroot (Jean Boussier)

Eregon (Benoit Daloze) wrote:

would also avoid compiling these duplicated Regexp again.

In theory yes, however my current patch doesn't go that far for simplicity's sake. However that would indeed be a nice followup or improvement.

#### #4 - 02/28/2020 11:44 AM - byroot (Jean Boussier)

From the developers meeting notes:

Preliminary discussion:

mame: Looks fine, if CI failure is fixed

ko1: The patch creates a regexp object, and then find already-existing instance. It is slightly suboptimal

ko1: Regexp is immutable in onigmo? If so, the optimization can be applied not only to literals but also to all regexp instances.

Discussion:

ko1: Is onigmo regex data structure immutable? If so, we can dedup all Regexp objects under lower layer (not only literals but also any regexp)

naruse: Yep. The key should be source string, encoding, and regexp options.

ko1: A simple reference count mechanism would be good enough

ko1: Regexp objects can be ignored

ko1: But there is no resource to develop... At least I cannot make effort for that

ko1: And his patch has some problems (mame: I failed to log)

akr: How about drop-in replacement of regcomp/regexec/regfree? (regcomp lookup a cache and count up the reference count. regfree count down the reference count.)

Conclusion:

ko1: I will propose the refernce count mechanism and tell the situation (I have no time to review and maintain the patch)

My opinion on this:

ko1: The patch creates a regexp object, and then find already-existing instance. It is slightly suboptimal

Yes, my plan was to improve that a bit later, I just see no point tackling this before I fix the CI failures.

is immutable in onigmo? If so, the optimization can be applied not only to literals but also to all regexp instances.

I don't have hard data, so I might be wrong, but:

- I highly doubt there's much duplication among dynamic regexps.
- If the registry is at this layer, it means it need to keep a copy of the src string. So it would increase each Regexp memory footprint.
- So I'm really not convinced it would actually be a win.

if CI failure is fixed

I'm circling around the issue, I have no idea if I'll be able to figure it out myself, but I've added some debug info on the PR, it might ring someone's bell.